Fault reasoning based on Naive Physics

Naveed Akhtar

Publisher: Dean Prof. Dr. Wolfgang Heiden

University of Applied Sciences Bonn-Rhein-Sieg,
Department of Computer Science

Sankt Augustin, Germany

April 2011

Technical Report 02-2011

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

Abstract

A system that interacts with its environment can be much more robust if it is able to reason about the faults that occur in its environment, despite perfect functioning of its internal components. For robots, which interact with the same environment as human beings, this robustness can be obtained by incorporating human-like reasoning abilities in them. In this work we use *naive* physics to enable reasoning about external faults in robots. We propose an approach for diagnosing external faults that uses qualitative reasoning on naive physics concepts for diagnosis. These concepts are mainly individual properties of objects that define their state qualitatively. The reasoning process uses physical laws to generate possible states of the concerned object(s), which could result into a detected external fault. Since effective reasoning about any external fault requires the information of relevant properties and physical laws, we associate different properties and laws to different types of faults which can be detected by a robot. The underlying ontology of this association is proposed on the basis of studies conducted (by other researchers) on reasoning of physics novices about everyday physical phenomena. We also formalize some definitions of properties of objects into a small framework represented in First-Order logic. These definitions represent naive concepts behind the properties and are intended to be independent from objects and circumstances. The definitions in the framework illustrates our proposal of using different biased definitions of properties for different types of faults.

In this work, we also present a brief review of important contributions in the area of naive/qualitative physics. These reviews help in understanding the limitations of naive/qualitative physics in general. We also apply our approach to simple scenarios to asses its limitations in particular. Since this work was done independent of any particular real robotic system, it can be seen as a theoretical proof of the concept of usefulness of naive physics for external fault reasoning in robotics.

Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
C.G	Center of Gravity
FOL	First-Order-Logic
KB	Knowledge Base
MD/PV	Metric Diagram/Place Vocabulary
NP	Naive Physics
QDE	Qualitative Differential Equation
QPT	Qualitative Process Theory
QR	Qualitative Reasoning
QSIM	Qualitative Simulation

1 Introduction

It is a common experience that faults occur even in the most carefully designed systems. Faults are even more common when the systems have to interact with real world. When an autonomous robot interacts with its environment, its performance can be degraded either by internal component malfunctioning or by external unseen circumstances. Although, diagnosis of internal faults in robotics is very important, but in many cases it is not sufficient to guarantee improvement in robot's performance. This is because unseen situations, which can degrade robot's performance, can exist even if the internal components of the robot work perfectly. Therefore, a robot should also be able to detect and diagnose the circumstances in which faults are external to itself.

In robotics, fault diagnosis typically requires tracking a very large number of possible faults in complex non-linear dynamic systems with noisy sensors. Usage of similar techniques for external fault diagnosis can result in enormous computational requirements. However, it can be noticed that we human beings also encounter faulty situations while interacting with our environment. And, we are able to reason about the situations and come to correct conclusion about the fault without explicit calculations or arithmetic models. We are able to do so even if we are not physics experts. Such use of *naive* physics concepts and *qualitative reasoning* for robots can also improve their ability to diagnose external faults with minimum computation.

In this work, we use *naive physics* knowledge for reasoning about external faults encountered by robots. The *naive physics* knowledge is common knowledge of physics novices and common people used for reasoning about everyday physical phenomena. We use *qualitative reasoning* on such knowledge to reason about external faults for robots. The reasoning process applies qualitative version of physical laws on properties of objects in robot's environment. The properties of objects are defined based on naive concepts behind them. In our approach we propose to divide properties and respective relevant physical laws based on different types of faults that can be detected by the robot. The underlying ontology of this division is based on studies conducted (by other researchers) on reasoning of physics novices about physical phenomenon.

This work can be seen as a proof of concept that naive physics is useful for external fault reasoning in robotics, despite many of its limitations. In this work, we also present brief critical reviews of some of the important works in the area of naive/qualitative physics for general understanding of naive/qualitative physics and its limitations. We use insights from these (and other such) works to develop our approach for robot fault reasoning. We exemplify the application of this approach by using it for different scenarios

involving simple manipulation tasks. The results and analysis of these experiments help in understanding the limitations of using naive/qualitative physics for fault diagnosis in general and using it with our approach in particular.

We present our work in nine chapters in this thesis. After this introduction, relevant background for understanding our approach is given in chapter 2. This chapter mainly comprises critical reviews of important relevant works in *naive physics* for AI. Followed by state of the art of fault diagnosis in robotics in chapter 3, chapter 4 briefly presents the context of our work in such approaches. This chapter also presents three simple scenarios for illustration of faults and reasoning about them. Chapter 5 discusses the crux of this work, where we propose our approach for fault reasoning and give relevant details. In the proposed approach we use properties of objects for reasoning. Chapter 6 presents a small framework that illustrates that how these properties should be defined in order to be used for our proposed scheme. In chapter 7 we give results and analysis of our approach based on the scenarios presented in chapter 4. We also state the limitations and assumptions of our work in this chapter. After brief discussion on some related works in chapter 8, we state conclusion and future directions of our work in chapter 9.

2 Background

In this chapter we describe relevant background for understanding this work. This section is mainly dedicated to understanding the concepts relevant to *naive/qualitative physics*, because this area is not well understood in general. Here we provide critical reviews of some important works related to *naive/qualitative physics* with emphasis on the concepts/ideas useful for understanding our work. The approach in this section is to highlight major problematic issues while dealing with *naive/qualitative physics*. We also describe some concepts of First-Order Logic (FOL) in this chapter. These concepts are limited to those which are crucial for understanding this work. A reader, completely new to FOL, can find a detailed account on FOL in Russell and Norvig [2002]. This section does not discuss concepts related to fault diagnosis. These concepts are described separately in chapter 4.

2.1 Naive physics

Hayes [1979] proposes development of a theory composed of entire knowledge of physics of naive reasoners in declarative symbolic form. Such theory, formed by logical formalization of everyday knowledge of physical world, is termed as *naive physics*. In the original proposal, the author mentions following four criteria/properties of formalization of such a theory.

1. **Thoroughness:** The formalization should cover the whole range of everyday physical phenomena.
2. **Fidelity:** It should be reasonably detailed.
3. **Density:** The ratio of facts to concepts should be fairly high. Such density is required in a formalism to pin down the exact meanings of the involved concepts.
4. **Uniformity:** There should be a common formal framework (language, system etc.) for the whole formalization.

Hayes proposes the structure of such a theory to be built around *clusters*, where a cluster is a linkup of *concepts* tightly related to each other by numerous *axioms*. Examples of such clusters are "shape, orientation and direction", "measuring scales" and "substance and physical states" etc. It is also proposed by the author to defer the implementation, application and inference strategy until the formalization of the knowledge is mainly complete. Since formalization of (almost) complete knowledge of everyday world is a huge

task, the author proposes that this research task should be carried out by a committee. The members of this committee should be assigned with a particular clusters to formalize. The author further proposes that, uniformity of the theory should be maintained by time to time meetings of the committee.

Some of the proposals in Hayes' work are similar to those of McCarthy [1968], however the idea of formalizing *naive physics* knowledge for reasoning, instead of common sense knowledge, is one of the major contributions of Hayes' proposal. The naive physics knowledge contains in it the common sense knowledge taken for granted while reasoning about a phenomenon (Hayes [1990]). This knowledge is sometimes also considered as knowledge of *physics novices* (Reiner et al. [2000]).

Misconceptions about Naive Physics

Davis [1998] mentions two major misconceptions found in the literature related to *naive physics* in AI[1]. These misconceptions have lead researchers to diverge from the actual proposal of Hayes and can also be seen among the reasons of downfall of popularity of *naive physics*. We briefly state these misconceptions here.

1. **Implementation and manipulation of knowledge:**
   It is a general misconception found in the literature that researchers (e.g. Kowalski [1979], Moore [1982] etc.) assume that a computer program which uses naive physics knowledge should explicitly manipulate logical formulas using some theorem proving method. However, in his work Hayes does not propose this. He proposes to choose FOL as representation language because it does not presuppose any particular form of implementation.

2. **Representation of spatial knowledge:**
   It can be seen in almost all the works that while representing geometrical information in FOL, basic spatial terms from natural language (e.g. right-of, left-of) are used as *the* primitives. However, in actual there are no such restrictions imposed by *naive physics* for spatial knowledge representation.

2.1.1 Difficulties with naive physics

"The Naive Physics manifesto" (Hayes [1979]) is one of the highly admired works in the field of AI, however it has not truly been followed as it was proposed (Davis [1998]). Researchers diverged from the original proposal to *qualitative physics*, which we describe in section 2.2. Here we summarize some of the difficulties that are generally faced while utilizing Hayes' idea of *naive physics* to accomplish any task.

[1]Here by "Naive Physics", we mean the original idea of naive physics as proposed by Hayes.

1. **No compilation of naive physics knowledge:**
   One key assumption in utilizing *naive physics* knowledge for any purpose is that we can reason (supposedly) correctly about physical phenomena with limited knowledge. However, correct reasoning requires that the structure of formalism is *dense*. But there is no such densely structured formalism available in the literature. The committees that were suppose to meet, never met and no theory was truly codified as proposed by Hayes.

2. **No absolute definition of naive physics knowledge:**
   It is not possible to have an absolute definition of a naive concept that can decide what exactly should be considered in a formalization. Therefore, it is not possible to develop any standard set of axioms which can relate concepts of naive physics.

3. **Describing shape and space is too difficult:**
   It is very difficult to describe knowledge regarding shape and space such that it can be used for reasoning in general.

4. **Level of generality of knowledge representation:**
   To formalize knowledge, it is important to choose the level of abstraction of underlying representation. If the concepts used in representation are too general then they may not serve the purpose of appropriate reasoning. If they are too specific then the formalism becomes too brittle and problem specific.

5. **Consistency of beliefs:**
   Hayes [1990] argues that *naive physics* knowledge includes the common sense knowledge taken for granted. This common sense knowledge comes in the form of beliefs of individuals. These beliefs can not be guaranteed to be consistent. Reiner et al. [2000] has shown many of such cases where beliefs of novices are inconsistent with reality. Although, some researchers (e.g. Vosniadou [2002]) are of the view that the beliefs of individuals can not be considered inconsistent but at the same time they argue that there is a constant element of learning involved with individual's beliefs.

2.1.2 Microworlds

One major variation in *naive physics* (in AI) from Hayes' proposal is the concept of *microworlds* (Davis [1998]). According to this concept, the knowledge is structured in the form of microworlds, where a microworld is an abstraction of a small part of physical interactions, sufficient to support some interesting collection of inferences. The major difference between this idea and Hayes' proposal is that instead of expressing a body of knowledge, this proposal focuses on justifying a collection of inferences. Accordingly, *any* theory that allows commonsense problems to be stated and solved will do in microworlds approach. In short, this approach is focused on developing specific physical theories.

Another major difference of this approach from Hayes' idea of *naive physics* is the way to treat the beliefs that are commonsensical but false. The author divides such beliefs into three categories given below, and allows the first two in any theory that can be utilized under this approach. The author also presumes that a cognitive model of naive reasoner can also include the third category in its theory.

1. Beliefs that are approximately correct in everyday context. For example, a moving ball will halt after some time even if no force is applied on it.
2. Logical consequences of (1). For example, a belief that if torque is applied to a gyroscope, it will rotate along the axis of the applied torque. This is not correct but is allowed as a consequence of (1) because objects generally behave like this but gyroscope is an exception.
3. Beliefs that are plain wrong, without either of above justification.

Although, microworlds approach changes the dimension of the theory from cognitive model (as proposed by Hayes) to competency theory, but still it suffers from many problems[2]. Among these problems following are worth reporting here.

1. Commonsense reasoning is not a task domain. It means that commonsense inferencing is just some module of some larger task. Therefore, it not possible to be sure about the decision that how commonsense inference should be formulated such that it serves the purpose of larger tasks.
2. There is no easy way to extend or integrate the microworlds.
3. There can be many microworlds but the approach by itself does not provide any guidance for choosing between them.

2.2 Qualitative physics

There are some differences of opinion in the literature about relation of *qualitative physics* and *naive physics*. For example Bratko [2001] states, "to emphasis the contrast between the *proper* physics taught in schools, and qualitative, commonsense reasoning about the physical world, the *qualitative physics* is sometimes called *naive physics*". On the other hand Davis [1998] argues that *qualitative physics* has a very different flavor as compared to the original notion of *naive physics*. In Davis' opinion *qualitative physics* "is algorithmic rather than declarative and is increasingly concerned with specialized applications rather than commonsense reasoning". Similarly, Forbus [2003] sees *qualitative physics* just as an adaption of *qualitative reasoning* (see section 2.3) focused on scientific and engineering problems.

[2]These problems mainly caused by the desire to combine microworlds into a large theory.

Our perception of *naive* and *qualitative physics* is in accordance with Davis' definition. It is also possible to draw a line between the *microworlds* approach and *qualitative physics* as per this definition. As we have discussed above that *microworlds* approach focuses on developing specific theories, on the other hand the works that fall under the category of *qualitative physics* focus on developing techniques which can be applied over different theories. Below we briefly review two important approaches in *qualitative physics* which are relevant for understanding this work.

2.2.1 Qualitative simulation

Qualitative simulation, is an inference process used in *qualitative physics*. The QSIM algorithm Kuipers [1986] is one of the major contributions in this area. The main idea of QSIM is to generate a *qualitative behavior* of a dynamical system (e.g. u-tube) in terms of a graph. This *qualitative behavior* represents values of *qualitative states* over time, where each *qualitative state* is a set of values of certain *parameters* of the system and their mutual *relations*. The *parameters* can take values from a small set of *landmarks*. QSIM uses an *abstraction* of ordinary differential equations called *qualitative differential equations (QDEs)* to model the system. These QDEs represent the *constraints* within the modeled system. The simulation process uses QDEs to predict the next qualitative state of system by enumerating the possible values and direction of change of system parameters. The graph of the behavior of the system is generated by predicting all possible values of the parameters.

2.2.2 Qualitative process theory

Qualitative process theory (QPT) (Forbus [1984]) uses the notion of physical process to define a theory for physical systems. These processes are the source of change in the system and this change is defined over extended time. A process in QPT is a structure that includes the objects on which it is applicable, quantity conditions, preconditions for the application of the process and its influences on other processes etc. Here again the system is defined by values of parameters (called quantities) which are taken from a collection of values called *quantity space*. In this approach the processes need to be defined before reasoning can be performed and addition of a new process at any time effects definitions of other processes.

2.3 Qualitative reasoning

Qualitative reasoning (QR) is the area of AI which creates representations for continuous aspects of the world, such as space, time, and quantity, which support reasoning with very little information (Forbus [2003]). In fact, QR is *the* approach used in *qualitative physics* for inferencing. On the other side, if one is aware of misconception (1) stated in section 2.1, it is easy to see that QR is not the only tool that can be utilized to get benefit from *naive physics* knowledge. Although there are many important issues related to knowledge representation and reasoning in QR [3], knowledge of which can be beneficial for understanding this work, but here we briefly state only few important issues related to reasoning about space and shape.

2.3.1 Reasoning about space and shape

To represent space and shape qualitatively one has to decide upon issues like ontology of the underlying representation, spatial relations, mereology (i.e. part-hood), directions, distance, orientation and many others. It is fairly hard to represent such aspects purely qualitatively such that the representation can be used for reasoning in an extensible manner. In fact, Forbus et al. [1991] says that *"no general purpose, purely qualitative representation of spatial properties exists"*. This notion is known as *poverty conjecture* in the literature related to QR. This conjecture is second by Cohn (Cohn and Hazarika [2001]), who argues that for spatial reasoning nothing weaker than numbers will do. The *poverty conjecture* is also the reason that many purely qualitative reasoning approaches can be found for restricted physical systems (Weld and Kleer [1990]), but it is hard to specify any such approach for spatial and kinematic mechanisms.

2.4 First-Order Logic

First Order Logic (FOL) is a logical language which is also known as first-order predicate calculus or predicate logic. It is a representation language of expressing knowledge that either subsumes other representation languages, like propositional logic, or forms the foundation of such languages e.g. higher order logics (Russell and Norvig [2002]). Knowledge described in FOL consists of expressions[4] that are composed of *constants* (e.g. `box, table`), *variables* (e.g. `X,Y`), *predicate* symbols (e.g. `place(O1, O2)`) and *function*

[3] A detailed account on these issues can be found in Forbus [2003].

[4] The expressions used for illustration use `Prolog` conventions rather than actual FOL conventions (e.g. variables start with capital letters instead of smaller). Same conventions are used in the complete thesis.

symbols (e.g. `place(O)`). The difference between *function* symbol and *predicate* symbol is that a *predicate* can take on values of either *true* or *false* when its argument gets instantiated, whereas on its instantiation a *function* may take on any constant as its value. For instance, `place(obj1, obj2)` can either be true or false, but `place(obj1)` may result in value `obj2`.

The symbols described above can be connected into *sentences* of FOL using *logical connectives* (i.e. 'or' $\vee$, 'and' $\wedge$, 'not' $\neg$, 'implication' $\Rightarrow$ and 'double implication' $\Leftrightarrow$). The variables in such sentences can be quantified using *quantifiers* (i.e. 'for all' $\forall$ and 'there exists' $\exists$). For example, if we want to state that "any object that is not stationary in an interval is moving in that interval", then we can state it by following sentence in FOL.

$\forall$ `Object`, $\forall$ `T moving(Object, T)` $\Leftrightarrow \neg$ `stationary(Object, T)`.

A collection of sentences like above serves as FOL *Knowledge Base* (KB) over which *inferencing* can be performed. Inferencing in FOL is performed using logical inferencing rules e.g. *unification, resolution, backward chaining* etc. These inferencing rules are applied to *symbols*. Therefore, arithmetic operations are not a part of logical inferencing. However, logical inferencing can be performed over *symbols* that are abstractions of arithmetic operations. For example, arithmetic summation can be abstracted in a *predicate symbol* as `sum(X,Y,Z)`. Which means that `Z` is the arithmetic sum of `X` and `Y`. Here `X`, `Y` and `Z` are variables which are allowed to get instantiated with *constant* symbols `pos, zero` and `neg`. The `pos` symbol represents any positive real number, `neg` represents a negative real number and `zero` represents '0'. The predicate `sum(X,Y,Z)` represents a *qualitative* abstraction of summation. It can be noticed here that this *qualitative* summation is *non-deterministic* (i.e. replace `X` with `pos, Y` with `neg` and you cannot tell that `Z` is `0, pos` or `neg`). Hence, a purely *qualitative* knowledge base suffers from loss of information at the hands of abstraction and the logical inferencing becomes less suitable for the applications where arithmetic operations are necessary.

3 State of the Art

Fault diagnosis and tolerance is one of the major challenges for robotics and AI community (Patton et al. [1989]). Experience has shown that even carefully designed and tested robots may encounter faults (Carlson and Murphy [2003]). Therefore, robotic fault diagnosis has always been an active area for researchers in robotics and AI. Fault diagnosis in robotics typically requires tracking a very large number of possible faults in complex non-linear dynamic systems with noisy sensors (Verma and Simmons [2006]). This makes model base diagnosis a useful approach for diagnosis. For example, Honghai Liu [2005] presents a model based approach called 'first priority diagnostic engine' that detects internal faults of robots by continuous monitoring of parameters of effectors and narrows down the fault to the component of the robot.

Mostly, fault diagnosis approaches in robotics deal with internal faults of robots. That is, the faults that are caused by malfunctioning of sensors or motors of the robot. For instance, the faults addressed in Verma et al. [2004] include mechanical component failures, such as broken motors and gears, faults due to environmental interactions, such as a wheel stuck against a rock, and sensor failures, such as broken encoders. Monteriu et al. [2009] presents a model based sensor fault detection and isolation system applied in real time to unmanned ground vehicles. Some approaches are mainly focused on tractability of the diagnosis. In one such approach Verma and Simmons [2006] takes advantage of structure in the domain and dynamically concentrates computation in the regions of state space that are currently most relevant without losing track of less likely states. Model based approaches are usually not considered good in graceful degradation[1]. Therefore, some approaches like Pettersson et al. [2007] also concentrate on model-free execution monitoring.

Qualitative approaches to fault diagnosis are considered successful, in general. For example, de Kleer and Williams [1987] uses model based diagnosis which infers behavior of composite device from knowledge of structure and function of its components and this inference is made qualitatively. Schroder [2002] proposes qualitative approach to fault diagnosis of dynamical system, mainly process control system. Similarly, Baniardalani et al. [2010] deals with qualitative model based fault diagnosis for processes. However, such qualitative approaches are mainly restricted to devices and well behaved processes. A review of qualitative model representations and search strategies used in fault diagnostic systems of processes can be found in Venkatasubramanian [2003].

[1] Simmon R., Fernandez J., Golden K., Joskowicz L., Pollack M., Model Based Monitoring and Diagnosis for Mobile Robots. http://www.cs.cmu.edu/rll/overview/reids02.

Qualitative approaches have also made their way into robotic faults diagnosis. For instance, LIU et al. [2005] presents a unit circle approach for qualitative modeling of the robot and implements a model in robot fault diagnosis. Similarly, Daigle [2008] presents a model-based approach to event-based diagnosis of hybrid systems based on qualitative abstractions of deviations from nominal behavior. Such approaches are also concerned with only the internal faults of the systems. Aside from fault diagnosis, *naive physics* is used in context of robotics by Kunze et al. [2011]. The authors translate naive physics problem to a parameterized simulation problem to get time interval based first order representation and use it for prediction in robot manipulation. Kunze et al. [2010] also presents a system to integrate commonsense knowledge into robot's knowledge base. These works emphasize on requirement of robot's human-like reasoning ability, which we achieve using naive physics for external fault reasoning.

4 Robotics faults

Broadly speaking, we can categorize robotic faults into two major types.

- Internal faults
  These are the faults which occur in the internal components of the robot and result into degradation of robot's performance. Example of such a fault can be, failure/-malfunctioning of actuator(s) or sensor(s).
- External faults
  These are the faults which occur in robot's environment despite perfect functioning of its internal components. For example, if a humanoid robot is required to place a spherical object on a table then there exists a possibility that the object rolls and falls on the floor (because of some reasons unknown to the robot). Such a behavior of the object (which was involved in robot's task), is an external fault that effects robot's performance.

In this work we are interested in *external* faults faced by a robot. Although, it is possible that an external fault itself is caused by some internal fault (e.g. the robot drops the object on the table from some height because of malfunctioning of its manipulator(s)), but for this work we assume that this is not the case. We assume that the internal components of the robot work perfectly and the occurred fault is indeed an external one. Furthermore, we are interested in reasoning about the faults which are *unknown* to the robot. These unknown faults occur due to unexpected circumstances in the environment.

4.1 Fault diagnosis

Diagnosis of a fault in a system consists of two major phases a) fault detection and b) fault isolation. For a system, the detection of a fault is to determine the occurrence of some abnormal event (i.e. fault) and the time of detection (R. J. Patton [2000]). This work assumes a priori detection of the fault. The second phase of isolation consists of determining the kind and location of the fault R. J. Patton [2000]. Using this diagnosis information the system can improve its performance.

In this work we consider model based diagnosis of unknown faults, shown in figure[1] 4.1. In such diagnosis of faults the system uses a model to predict its behavior or the behavior

[1]Figure taken from, ' Kuestenmacher A.(2010) Presentation: Diagnosis of unknown faults.'

of the environment and uses this prediction to detect the fault. In our case the robot uses the model of the world to predict the correct outcome of the task it needs to perform. After the completion of the task the robot compares its new observations to the prediction (which comes from the effects of the planning operators used by the robot in performing the task). If the prediction is in contrast with the observations then the robot needs to diagnose the fault. In case the fault is known to robot then it can take relevant actions to overcome it. If the fault is unknown, then the robot needs to reason about it such that the outcome of reasoning can help in improving its model. In this work we are mainly concerned with the reasoning part of the diagnosis (shown in 'red' in figure 4.1) and we use *naive physics* knowledge for reasoning.

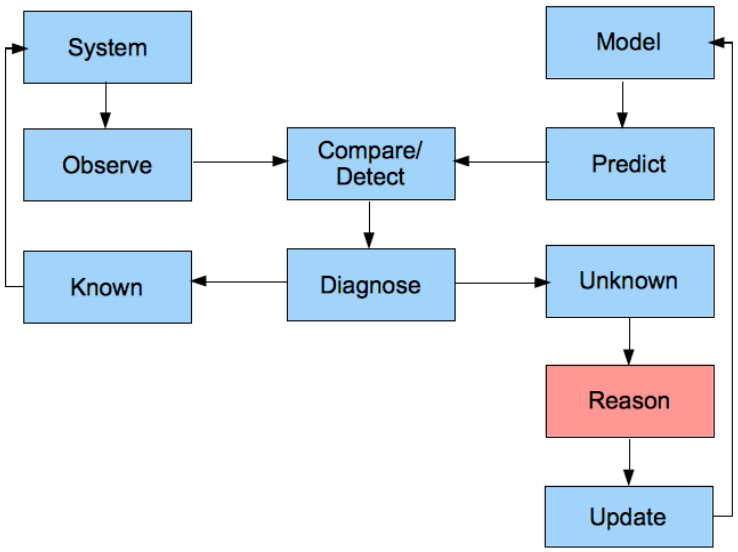

Figure 4.1: Model based diagnosis for unknown faults

It can be noticed that since we are dealing with external faults, the isolation phase (as stated above) of the diagnosis will not locate the fault within robot's components. The fault isolation will result in producing some *hypotheses* which help in isolating the conditions in the external environment which cause the fault. Construction of such hypotheses can only result from incorporating the behavior of the involved object(s), over extended time, in reasoning. Thus, reasoning about the fault also includes the "identification" (R. J. Patton [2000]) of the fault (i.e. determination of size and time-variant behavior of the fault).

4.2 Scenarios

In this section we briefly describe three scenarios in each of which a humanoid robot performs a simple task. These scenarios are chosen to illustrate the occurrence of external faults and corresponding working of the approach proposed in (upcoming sections of) this work. Each task involves manipulation of some object to achieve a goal.

4.2.1 Scenario I

In this scenario, as shown in figure 4.2, a (NAO [2]) robot is trying to place a dice (i.e. a cube) on a table (i.e. larger cube). While doing so the robot releases the dice in a position that finally results in falling of the dice on the floor. The situation shown in figure 4.2 is actually the result of wrong grasping by the robot when it picks up the dice from the floor. This wrong grasping in turn is the consequence of a slight push to the dice by the robot itself when it tries to pick the dice. Although, all the actuators and sensors performed perfectly fine in this scenario but the fault occurs. This is an example of *external* faults, which cannot be located within the components of the robot.

Figure 4.2: Putting an object on table.

The detection of this fault, by the robot, is accomplished by comparing the *effects* of the actions performed by the robot with the actual observations after the completion of the task. That is, the robot uses its model (in planning module) to predict that the dice is on the table but the observations say that dice is on the floor. Therefore, the robot is able to detect that some fault has occurred.

[2] http://www.aldebaran-robotics.com/

4.2.2 Scenario II

In the second scenario we consider a robot dropping an object (i.e. toy duck) into a container (i.e. a basket). Figure 4.3 shows this scenario in which the robot is about to release the object in the correct manner such that the object drops into the container. The robot performs the task by detecting the container and dropping the object above it. Here, it is possible that the container is already filled to the top and the object falls out on the floor because of that. It is also possible that top of the container is covered with a lid and the robot does not have the ability to detect it and it is unable to complete its task. Similarly, there can be many other circumstance which can result into occurrence of the fault in which the object falls out on the floor. Such an external fault is again detected as in scenario I.

Figure 4.3: Putting an object into a container.

4.2.3 Scenario III

In this scenario the robot picks up an object (i.e. a bottle) from the table. In figure 4.4, the robot is able to do it correctly. However, it is also possible that the object is not placed on the table correctly or the object is so light that the initial touch from the robot makes it fall from the table and the robot is unable to complete its task because of this external fault.

Figure 4.4: Picking an object.

All three scenarios mentioned here involve simple manipulation tasks but completion of these tasks can suffer from many external faults. These faults can occur even if the sensors and actuators of the robot work perfectly fine. Existence of such faults is possible even without any external agent. Furthermore, these faults are unknown to the robot because the circumstance which have lead to the faults are unseen and the robot is unaware of such situations.

5 Fault reasoning using naive physics

In previous chapter three simple scenarios are described to illustrate external faults. We can make following important observations from these scenarios.

1. In all three scenarios the occurred faults are related to *location* of some object. That is, the goal of the robot is to achieve a certain location of the object but the external fault does not allow it to do so.

2. Occurrence of the fault can only be detected by the robot *after* it completes its action and this occurrence causes the object to go into a state where it is stable/stationary after a while[1].

3. Although any fault occurs at a particular instant of time, but the faulty behavior of the object is extended over time. Thus, it is more natural to attribute a fault to a state that is extended over time rather than to a state that is represented at an instant of time. Associating the fault to an instant may require tremendous amount of information because the behavior of the object can evolve in many different ways from a state at an instant.

4. Replacing the object in each scenario, can result in different circumstances. For instance, if the dice in scenario I is replaced with a smaller dice then the situation in figure 4.2 may not result into an external fault.

5. Changing the attributes of the same object can also result into different circumstances. For example in scenario III, if the quantity of the liquid inside the bottle is changed, it can behave differently. That is, an empty bottle can easily fall as compared to half filled bottle if it is not correctly picked/released.

Based on these and other such observations we develop an approach for external fault reasoning in this chapter of the thesis. This approach/mechanism uses *qualitative reasoning* to find the reasons of the occurred fault and it utilizes *naive physics* knowledge for reasoning. We present the *use case diagrams* for the developed approach in section 5.1. These diagrams introduces the reader with relevant behaviors of the robot and the fault reasoning system. In section 5.2, we describe detailed working of the approach by explaining the schema of the mechanism.

[1] Without the presence of an external agent.

5.1 Use case specification

The robot acts autonomously in its environment and the fault detection and diagnosis is done by its own internal components. Therefore, the diagrams shown in this section show the internal components of the robot as *actors*. The relevant *behaviors* of the system are shown as use cases. The diagrams are given in 'general to detailed' sequence. That is, diagram '1' is showing the most general behaviors and actors whereas diagram '3' shows the details of behaviors and actors according to the developed approach. We give *brief* descriptions of use cases from diagram '3' in appendix D.

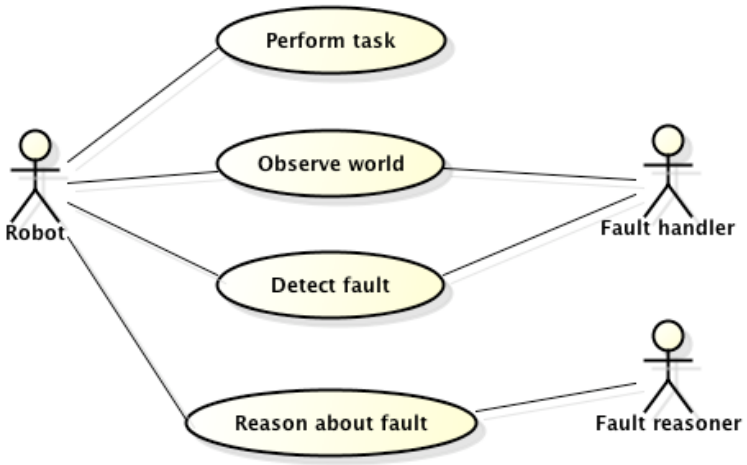

Figure 5.1: Use case diagram 1

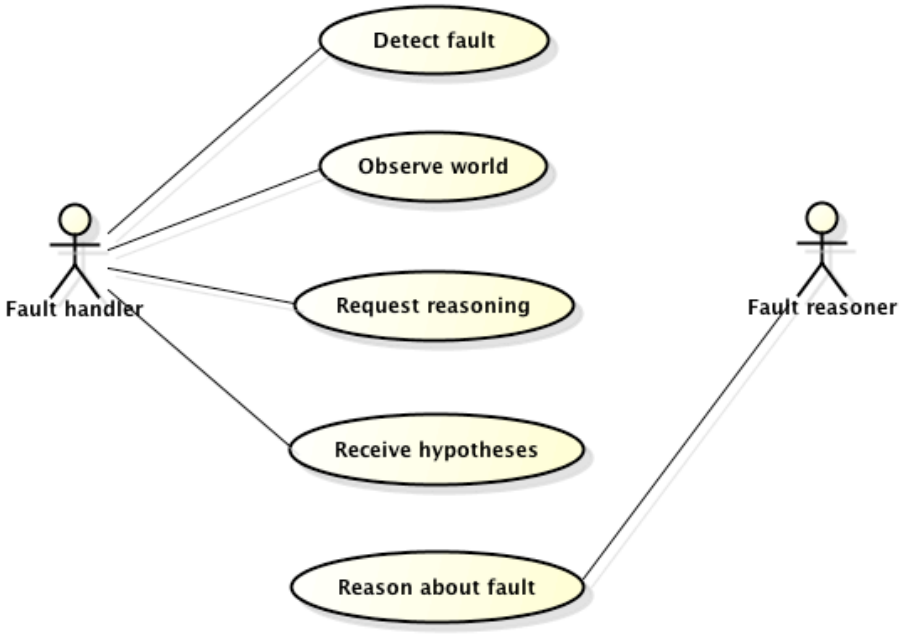

Figure 5.2: Use case diagram 2

Figure 5.1 shows that the robot interacts with the system (i.e. its world) by performing tasks and observing it. It also detects and reason about any fault occurring in its environment. The actors on the right hand side of use cases are the components *within* the robot which exhibit the behaviors associated with them. Behaviors of these components are further refined in figure 5.2. The *fault handler*, is assumed to handle the fault by detecting it and requesting for reasons of their occurrence from the reasoner. This is shown in diagram '2', where the *fault handler* that interacts with the world by observing it, is shown on the left side of the use cases.

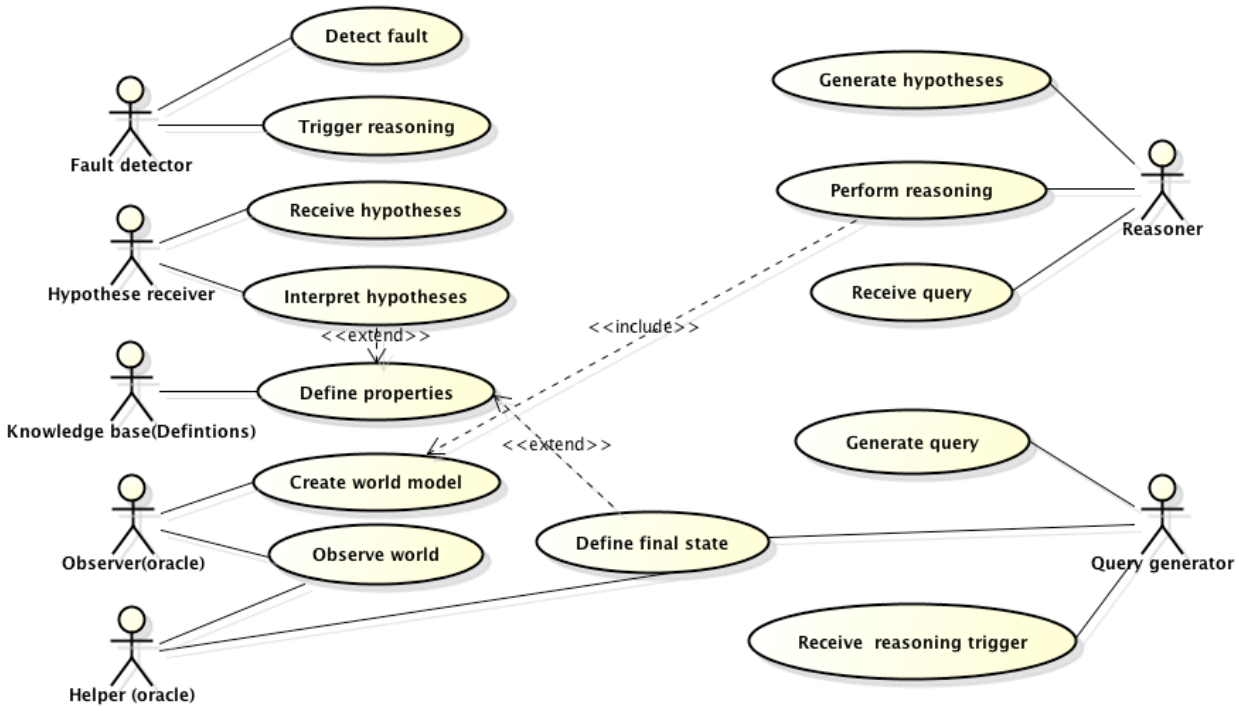

Figure 5.3: Use case diagram 3

Diagram '3', in figure 5.3, shows further refinement of behaviors and actors. The *fault handler* is broken down into five actors and the *fault reasoner* is shown as two different actors. The reader is reminded here that the actors are assumed to be (software) components within the robot whose associated behaviors are the functions performed by them. We only give brief description of use cases from diagram '3' in appendix D. Detailed description and explanation of the diagrams are omitted because section 5.2 explains the overall approach/mechanism in detail, which includes all relevant explanation. Use case diagram '3' can be better comprehended after reading the next section of the thesis.

5.2 Schematics for reasoning

For reasoning purpose we perceive the robot's world and occurrences of the (external) faults as following.

- The goal of the robot is to achieve a particular *state* of its world. In case the goal is achieved there is no occurrence of (external) fault.
- If some (external) fault is detected, it is because the goal state is not achieved *perfectly* and it is degenerated into some *final state* of the world, such that the *final state* is different from the actual *goal state*.
- The reason of the fault lies in the *imperfect* goal state (i.e. the state which degenerated into the *final state*) and this imperfect state can be one of many possible states which can result into the *final state*. We call all these possible states as *intermediate states*.
- Since there is no external agent involved in the occurrence of fault, the change from any *intermediate state* to the *final state* is a result of some natural *physical phenomenon* (e.g. gravity, friction, air pressure etc.), the source of which is unknown to the robot.
- The actual reason of the fault is found when a list of possible *intermediate* states is generated that contains the actual *imperfect goal state* and meaning of this state is understood by the robot.

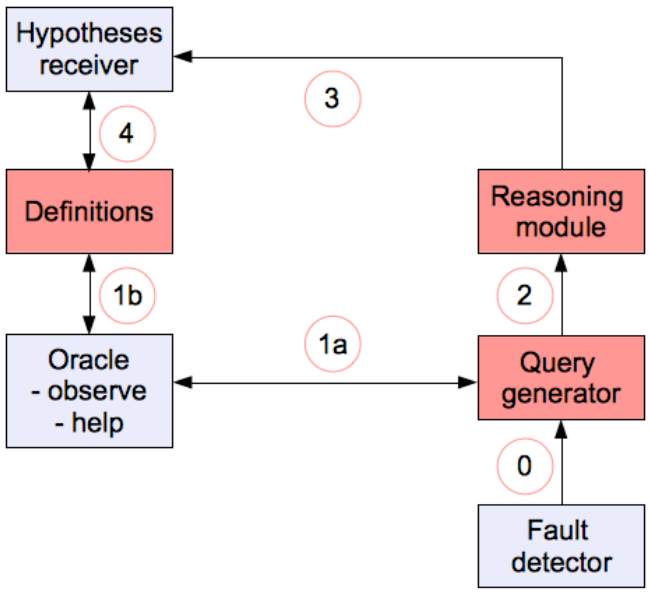

Figure 5.4: Schematic diagram

Based on this perception of fault occurrence and diagnosis, the schematics of fault reasoning is shown in figure 5.4. The complete mechanism of fault reasoning proceeds in four steps after the detection of the fault. In first step, a set of *queries* (regarding the reasons of occurred fault) is generated by the *query generator* with the help of an *oracle*. In second step, this set of *queries* is posed to the *reasoning module* which creates answers to the queries by forming hypotheses composed of possible *intermediate states*. In third step, the list of hypotheses is given to the *hypotheses receiver*, which is (assumed to be) able to choose the best hypothesis. Lastly, the *hypotheses receiver* interprets the meanings of the hypothesis based on the definitions used to create the queries.

In figure 5.4, all the components of the schematics except *'oracle' and 'definitions'* assume that robot's world consists of well defined *objects*. Which means that all the reasoning is done by considering an object as a primitive entity. The reason for such a high level of abstraction is twofold.

- Firstly, the detection of the fault is assumed to be performed using literals of planning module (not shown here) which operates at the same level of abstraction.
- Secondly, we use *qualitative reasoning* for diagnosing the fault and *qualitative reasoning* deals with symbols. Using objects as constant symbols results in better inferencing.

Below we give details of working and rationale of the approach used for *'query generator'* and *'reasoning module'*. The *'definitions'* component is the topic of discussion for next chapter. The components shown in 'blue' color are out of the scope of this work, hence those are only briefly discussed where necessary.

5.2.1 Query generator

When the *fault detector* detects the fault, it triggers the *query generator*. This trigger signal includes in it the *type* of the fault that has occurred. This *type* is directly decided from the *literals* in the planning operators which indicates the occurrence of the fault. Since this work is done independently from the planning module of the robot, here we assume that possible *types*[2] of fault fall under following three major categories.

- *Location:* of the object involved in the task (e.g. faults discussed in section 4.2).
- *Shape:* of the object (e.g. breaking/disintegration of an object.).
- *Movement:* of the object or robot (e.g. stoppage of robot or some part of it due to glass door etc.).

[2]The assumed types are likely to cover many faults, however it can neither be claimed complete nor correct (or wrong). The correct and complete list of such types can only be constructed based on allowed relations in the planning module of the robot for which the fault diagnosing system is being developed. Refinement of the 'types' shown here is expected in real application.

An appropriate *query* for the *reasoning module* consists of *final state* of the involved object, a probable *physical phenomenon* (e.g. gravity) that can cause the fault and expected correct value of the variable (i.e. property) which was not achieved. In our approach, a *state* of an object is characterized by some of its *properties*. These *properties* are finite and relevant for reasoning about the particular *type* of the fault.

Substance schema

The properties used to describe the state of the object are primarily derived from the substance schema proposed by Reiner et al. [2000]. This schema is developed by the authors (of Reiner et al. [2000]) based on many studies conducted on inferencing of physics novices about daily life concepts. The left column of the table 5.1 shows the properties of substance used by the physics novices to reason about physical phenomenon. The right column shows the meaning of these properties. The shown properties are also used by physics novices to reason about concepts which are new to them (e.g. forces, light and heat etc.).

Properties	**Meanings**
Locational	Have definite location
Pushable	Able to push and be pushed
Frictional	Experience drag when moving along a surface
Containable	Able to be contained by something
Consumable	Able to be usedup
Stable	Do not spontaneously appear or disappear
Corpuscular nature	Have surface and volume
Additive	Can be combined to increase mass and volume
Inertial	Require force to accelerate
Gravity sensitive	Fall down when dropped

Table 5.1: Substance schema (Reiner et al. [2000]).

Ontology for fault reasoning

To reason effectively, the *reasoning module* requires to deal with only a small number of relevant properties of the object. The *query generator* is also required to enumerate these properties in its queries. For that matter, the *query generator* utilizes an ontology of *properties* which are associated with *physical phenomena* that can cause a particular *type* of fault. This ontology is shown in figure 5.5. The properties used here are mainly derived from the properties and concepts used in substance schema of Reiner et al. [2000]. For actual practical application of our approach this ontology may require to be refined.

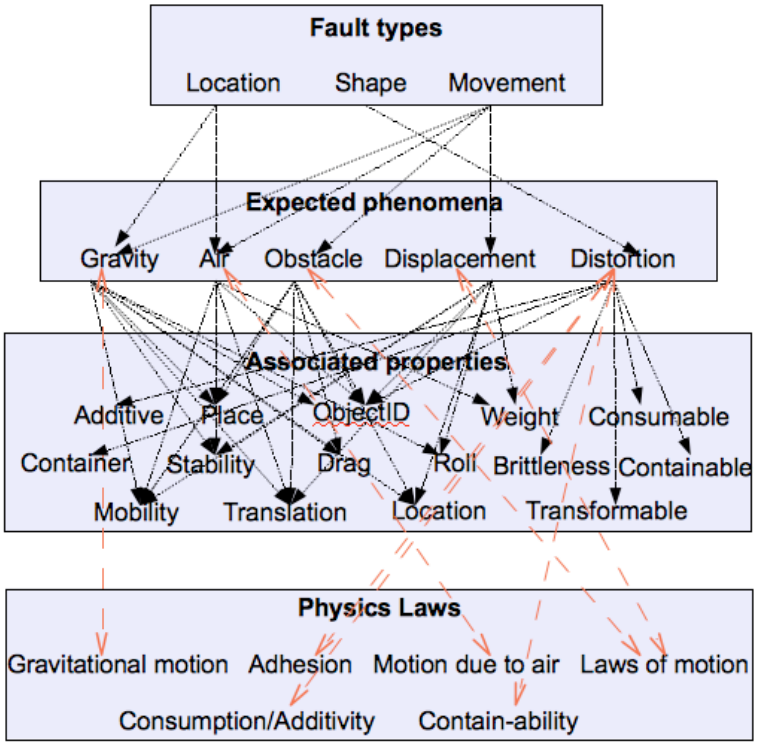

Figure 5.5: Ontology for fault reasoning

The role of 'Oracle'

Using the inbuilt ontology, the *query generator* asks for the *oracle's help* to fill in the values of relevant properties of the object in its *final state*. The *oracle* possesses the knowledge of the world through *observations*. Based on these *observations* and the *definitions* of properties in the *'definitions'* component, the *oracle* fills in the values of properties. The complete process of *query generation* is shown as step 1*a* and 1*b* in figure 5.4. The `prolog` code for *query generator* module can be found in appendix-B.

5.2.2 Reasoning module

After receiving the relevant queries, the prime function of the *reasoning module* is reduced to apply the *physical laws* to trace the possible *intermediate states* from the *final state* of the object. For different *types* of fault different *physical laws* are to be applied on different *properties*. For each *type* of fault, we associate the relevant *physical laws* with respective *physical phenomenon*. This association is shown with 'red' arrows in figure 5.5. When a *query* is posed to the *reasoning module* it already contains the relevant *properties* and the information regarding *physical phenomenon*, and hence the *reasoning module* selects the correct law to be applied on relevant *properties*. It can be noticed here that the complete ontology shown in figure 5.5 is actually used by both *query generator* and *reasoning module*.

The *reasoning module* uses logical inferencing to find the possible *intermediate states*. This inferencing utilizes a model of the world which consists of following information.

- Objects in robot's world.
- Numeric values for some properties (e.g. height, weight)
- Possible values (i.e. in terms of symbols) of properties.
- Relations between objects (e.g. near, far).
- Auxiliary definitions (e.g. `same`, `different`).

Based on this information and *physical laws*, the reasoning module generates a set of *hypotheses*. Each *hypothesis* in this set is a possible *intermediate state* that can degenerate into the final state because of the fault occurrence. This set is transmitted to the *hypotheses receiver* (shown as step '3' in figure 5.4). The *hypotheses receiver* is assumed to select the best (or a couple of good) hypothesis(es). The selected hypothesis(es) is interpreted by the *hypotheses receiver* according to the definitions in the *'definitions'* component. The `prolog` code used for *reasoning module* in this work can be found in appendix-C.

6 Definitions of properties

The *query generator* and the *reasoning module* in chapter 5 use different properties of the object (e.g. *stability*, *mobility*) to find the possible *intermediate states*. These modules assume that useful definitions of these properties are available to the robot through a *definition* component. These definitions are also required to get the exact meaning of the diagnosis at a lower level of abstraction. In this chapter we examine that how these definitions can be formed. We do it by analyzing the requirements for any possible option and then using these insights to establish a small framework. This framework defines some properties required in *scenario I* of section 4.2.1.

6.1 Requirements for defining properties

The level of abstraction of the properties utilized by *query generator* and *reasoning module* is very high. It means that the definitions must be such that they cover as many objects and situations as possible. However, in real world these definitions depend (somewhat closely) upon circumstances and objects under consideration. For example, a dice made of a certain substance may remain *stationary* on an inclined surface in presence of gravity and friction only, whereas a sphere made of same substance and having same weight may be *moving* on the same surface. Similarly, both the objects can exhibit similar properties regarding their *translation* if the inclination of the surface is reduced or increased. Observation (4) and (5), in chapter 5 also indicate the same dependency between objects and their behavior (i.e. properties) under particular circumstances.

Normally, an engineer or a scientist can employ newtonian mechanics to decide upon the *translation* of the objects considered in the example above. Newtonian mechanics utilizes numerical values of certain parameters (e.g. slope of the surface, frictions coefficients and weights etc.) to find the correct results. Since we need to consider mechanics and kinematics to decide upon such trivial properties of the object, it is quite apparent that the definitions of these properties can not be purely qualitative (as per *poverty conjecture* Forbus et al. [1991]). On the other hand, it is also not possible (or at least not feasible) to build *Metric Diagram/Place Vocabulary (MD/PV)* representation (Forbus et al. [1991]) to define properties because our problem domain is real world with endless possibilities. Furthermore, it can be seen that certain definitions of properties may depend upon each

other (e.g. an *unstable* object may also be a *moving* object). Therefore, it is also required that definitions are mutually consistent.

Here, we need to remind ourselves that the intended definitions of the properties need not be perfect according to respective concepts in mechanics and dynamics etc. This is because the properties considered here exist because of naive concepts in the first place. Therefore what we actually need, are the explanations which satisfy the naive concepts of *translation*, *stability* and *mobility* etc. Although, these explanations can not be entirely independent from newtonian mechanics, however the naive version of these concepts can have much lesser dependence on numeric values.

A closer look at the properties mentioned in table 5.1 and figure 5.5 reveals that the definitions of some properties need an extended time in order to correctly represent the concept behind them. For example, the concept that an object is *moving*, can only be described using an interval and not just an instant. On the other hand, there are also some other concepts (e.g an object being *stationary*) that can be described at an instant only. This fact leads towards the requirement of notion of time in the definitions. Such time dependence of definitions is also in accordance with observation (3) in chapter 5.

From the discussion above it is clear that the definitions of properties require utilization of a framework that formalize these definitions in a way a common person thinks about them (i.e. not strictly following newtonian mechanics). However, all these definitions must be consistent with each other. Furthermore, this framework should utilize as less quantitative information as possible and this information should preferably be object independent. This would enable us to define properties with lesser information and such definitions have better chances to be general or very loosely dependent on circumstances. Logic is a preferable representation for such a framework, since the definitions explain naive concepts. In our settings a logical framework would also serve the purpose of lowering the level of abstraction of the meanings of the *intermediate states* found by the *reasoning module*.

6.2 A framework

In this section we formalize a small logical framework with an intent to exemplify that how it can be done such that it suffices the requirements of definitions for our settings. It should be noted that this framework is only primitive and its completeness is not claimed here. We mainly illustrate the approach to handle the difficulties faced while developing useful definitions for our mechanism. Furthermore, the framework is only meant for solid objects. We take some basic concepts, related to defining the geometry of the domain, form Davis

[1986]. However, the approach differs greatly from Davis's work in dealing with time and physics itself. We consider the geometry to be $\mathbf{R}^3$, a subset of which is occupied by any object in the domain. Also, the shape of any object is equal to the closure of its interior and the object itself is considered as a primitive entity.

The notion of time is captured by the concept of *intervals* and *instants*. An *interval* is just a set of *instants*. Ideally, an *interval* consists of infinite *instants*, but here we consider this number to be finite and small for practical reasons. Those *predicates* or *functions* which depend on *intervals* or *instants*, have explicit mentioning of respective temporal notions in their definitions. Others, which do not explicitly mention time are atemporal and are mostly related to describing geometry of the world. The *definitions* presented here are used in chapter 7, however we repeat the relevant *definitions* there (where necessary) in plain english language. Therefore, a reader not interested in deeply understanding the logical definitions can skip them in first reading and return to relevant definitions when those are referred in later parts of the thesis.

Conventions

Following conventions are used in the definitions of the framework.

- Definitions of *functions* and *predicates* are first stated in simple english. Logical form of the definition are given only for those *predicates* which depend upon other definitions. Axioms are also stated in logical form.
- Quantifiers are only mentioned where they need to be emphasized. Otherwise, universal quantifiers are omitted.
- First letter of any *variable* is kept capital whereas the first letter of *constants* is kept small.
- Name of *constants* exactly state their meanings (e.g. `floor`) whereas meanings of *variables* are stated in respective definitions. A double letter variable denotes a set of a single letter variable. That is, `T` shows an *interval* and `TT` shows a set of *intervals*. Similarly `X` shows a point in $\mathbf{R}^3$ and `XX` shows a set of such points. Numeric values in the names of variable are used only to distinguish them from same kind of variables.

Assumptions

Following are the major assumptions made in the definitions.

1. An object consists of an interior and a boundary. Where the boundary itself consists of surfaces. These surfaces are distinguished by discontinuities in the boundaries (i.e. edges).
2. An object has at least two surfaces which correspond to its top and bottom. If the object has only one continuous surface then it is broken into two as top and bottom.
3. All surfaces are bounded except the `floor` which is also not inclined anywhere.
4. There exists a fixed external frame of reference in which z-axis has value zero at floor which increases positively in straight upward direction.
5. There exists a special constant *up* which represents a (hypothetical) line starting from floor and moving straight up till infinity, parallel to z-axis of reference frame.
6. A *function* takes on only one value. If a *function* is defined over an *interval* such that it takes on different values in the same interval, then we assume availability of heuristics to resolve the situation.

Below are the definitions of *predicates* and *functions* forming this (incomplete) framework. We only state those *axioms* which will be useful in later part of this report.

Definition 6.2.1: Predicate `place(Object1,Object2)` is true when `Object1` is a place for `Object2`.

`place(Object1, Object2)` $\Leftrightarrow$
$\exists_T[$ `surfaceArea(top(Object1, T))` $>$ `surfaceArea(bottom(Object2, T))`
$\wedge\neg$ `container(Object1, Object2)` $]\vee$ `container(Object1, Object2)`

Definition 6.2.1a: Predicate `place(Object1, Object2, T)` is true when `Object1` is the place for `Object2` in the complete interval `T`.

`place(Object1, Object2,T)` $\Leftrightarrow$ `on(Object1, Object2, T)`

Axiom 6.2.2: `Floor` is a place for every object.

$\forall_{Object}$ `place(floor, Object)`

Definition 6.2.3: Function `top(Object, T)` maps the `Object` to the surface farthest from the floor in interval `T`.

Definition 6.2.4: Function `bottom(Object, T)` maps the `Object` to its surface that is nearest to the floor in interval T.

Definition 6.2.5: Function `surfaceArea(S)` maps a surface of an object to its area.

Definition 6.2.6: Predicate `stationary(Object, T)` is true if the `Object` is stationary in the interval `T`.

`stationary(Object, T)` ⇔
∀ `I` ∈ `T` `coordinates(V, centerGravity(Object), I)` ∧ `V = constant`

Definition 6.2.7: Function `centerGravity(Object)` maps an `Object` to its center of gravity.

Definition 6.2.8: Predicate `coordinates(V, X1, I)` is true when vector `V` gives the coordinates of point `X1` at instant `I`.

Definition 6.2.9: Predicate `moving(Object, T)` is true when the `Object` is moving at any instant in the interval `T`. In other words the `Object` is moving if it is not *stationary* at all the instants in the interval `T`.

`moving(Object, T)` ⇔ ¬ `stationary(Object, T)`

Axiom 6.2.10: An object can only be either `stationary` or `moving` in the interval. It can not have both states in the same interval.

∀ `T` ∈ `TT` ¬[`moving(Object, T)` ∧ `stationary(Object, T)`]

Definition 6.2.11: Predicate `fixed(Object)` is true when the `Object` is permanently fixed at its place.

`fixed(Object)` ⇔ ∀ `T` ∈ `TT` `stationary(Object, T)`

Definition 6.2.12: Predicate `moveable(Object)` is true when the `Object` can move at any time. In other words it is not fixed in all the intervals.

`moveable(Object)` ⇔ ¬ `fixed(Object)`

Axiom 6.2.13: An object can only be either `fixed` or `moveable`. It can not be both.

¬[`fixed(Object)` ∧ `moveable(Object)`]

Definition 6.2.14: Predicate `stable(Object, T)` holds true when the `Object` is stable in the interval `T`.

`stable(Object, T)` ⇔
`on(Object, Object1,T)` ∧∀ `I` ∈ `T` [`coordinates(V, centerGravity(Object),I)` ∧
`parallel(make-line(V, V1), up)` ∧ `V1` ∈ `XX = top(Object1,T)`]

Definition 6.2.15: Predicate `unstable(Object, T)` holds true when the `Object` is unstable in the interval `T`.

unstable(Object, T) ⇔
on(Object, Object1,T) ∧∃ I ∈ T [coordinates(V, centerGravity(Object),I) ∧
parallel(make-line(V, V1), up) ∧ V1 ∉ XX = top(Object1,T)]

Axiom 6.2.16: An object can only be either stable or unstable in an interval. It can not have both states in the same interval (according to the framework).

∀ T ∈ TT ¬[stable(Object, T) ∧ unstable(Object, T)]

Definition 6.2.17: Function distance(X1,X2) maps two points X1 and X2 onto distance between them. Since the points are given as coordinates in 3D space therefore they can also be considered as vectors.

Definition 6.2.18: Predicate parallel(V1,V1) holds when vectorsV1 and V2 are parallel.

Definition 6.2.19: Function make-line(X1,X2) gives a straight line between points X1 and X2.

Definition 6.2.20: Predicate on(Object1, Object2, T) is true when Object1 is on Object2 in interval T.

on(Object1, Object2, T) ⇔
touch(bottom(Object1,T), top(Object2,T), XX)

Definition 6.2.21: Predicate touch(S1,S2,XX) is true when surface S1 and S2 intersect each other and XX is the set of points from their intersection.

touch(S1,S2,XX) ⇔
surface(S1) ∧ surface(S2) ∧[S1 $\bigcap$ S2 = XX $\neq \phi$]

Definition 6.2.22: Predicate rollable(Object) is true when the Object is rollable.

rollable(Object) ⇔
∃ T [∀ I ∈ T moving(Object, T) ∧ on(Object, Object1, T) ∧
coordinates(V, centerGravity(Object), I) ∧ parallel(make-line(V, V1), up) ∧
V1 ∈ XX = top(Object1, T) ∧ distance(V, V1) = constant]

Axiom 6.2.23: A movable Object can either be rollable or drag-able.

∀ T ∈ TT movable(Object) ⇔ ¬ dragable(Object) ∨¬ rollable(Object)

Axiom 6.2.24: A nonrollable object is dragable.

nonrollable(Object)⇔ dragable(Object)

Axiom 6.2.25: A nondragable object is rollable.

nondragable(Object)⇔ rollable(Object)

As discussed above this framework is not developed with the intent to be complete. This is why only those definitions and axioms are presented here which are useful in understanding the application of the schema, shown in figure 5.4, to the scenarios given in section 4.2. However, following observations are worth noting in the definitions.

1. The presented definitions are mutually consistent. That is, no two *predicates* that represent opposite concepts can be true in the same interval (or set of intervals).
2. Definitions *do* represent redundant information. Which means that the frame work does not care that an *unstable* object (in an interval) is also a *moving* object etc.
3. The definitions are biased towards *location* type faults. That is, mostly the definitions are only utilizing the information that will result into change of their truth value with the change in location of the object. In other words the definitions are *not* truly correct in general.
4. The definitions are independent from objects and circumstances.
5. We allow different definitions to refer to the same object/concept (e.g. definition 6.2.1 and 6.2.1a).

7 Results and analysis

In this chapter we present results of application of the proposed approach on the scenarios shown in section 4.2. We mainly focus on scenario I and provide a complete overview of the application and results of the approach. We use scenario II and III to analyze the approach for its extensibility and limitations. These limitations, along with other possible difficulties of applying this approach in general, are given in section 7.3. We enumerate the important assumptions made in this work in section 7.4 of this chapter.

7.1 Scenario I

Consider scenario I, shown in figure 4.2. This scenario occurs in robot's world that has different *objects* in it, e.g. a `dice`, a `table`, `floor`, a `chair`, a `shelf` and a `paper`. Only the `dice` and the `table` are visible in the figure, among these objects. Assume that at the instant of occurrence of fault the `chair` is placed *near* the `table` and the `shelf` is standing *far* from it. To start with, we do not consider presence of the `paper`. When the fault has occurred the `dice` has fallen on the `floor` and the robot detects the fault because some of the effects of its last action are not achieved. For example, a predicate `on(table,dice)` is not true after the completion of the action.

In this work we assume that the *fault detector* is able to point out the *type* of the fault from the unachieved effects. Here the relation `on/2` is a clear indication that the fault is of *type location*. The *fault detector* sends following signal[1] to the *query generator*.

```
fault(location).
```

Based on the *type* of the fault, the query generator enumerates the properties relevant for reasoning about this *type* of fault. These properties need to obtain (*symbolic*) values to represent the `dice` in its *final state*. These values are obtained through *oracle's* help. In this scenario, the *query generator* asks for values of following properties.

```
[objectID, place, stability, translation, mobility, rollability].
```

In addition to this the *oracle* is also asked to state the expected location of the *object* if there were no fault. This information is easily available through the failing relation (i.e. `on(table, dice)`).

[1] The syntax is compatible to the code given in appendix B.

The *oracle* also *observes* the world. Here we assume that the *oracle* is able to see the *object* in its *final state*. Based on its observations, its knowledge from the failing relation and the definitions from the *'definition'* component, the *oracle* fills in the values of properties as following.

- Since the *object* involved in the failing relation is `dice`, therefore `objectID` gets the value `dice`.
- Since the oracle observes that the *object* keeps a steady position (in an entire interval) *on* the floor after it has fallen, therefore *definition 6.2.20* is true when `Object1` is instantiated with `dice` and `Object2` with `floor`. Because of this, *definition 6.2.1a* is also true with the same instantiations in the same interval. Hence the value of `place` is `floor` according to *observations* and *definitions*.
- To get the value for `stability`, the oracle checks the truth values of definitions *6.2.14* and *6.2.15*. Since the `dice` is *on* the `floor` in the observation *interval*, as per definition *6.2.20*, and at all the *instants* in the *interval* the center of gravity of the `dice` does not leave the *top* of the `floor`, therefore the `dice` is *stable* in the *interval*. Hence from definition *6.2.14* `stability` gets the value of `stable`.
- In the observation *interval*, center of gravity of the `dice` never changes its position. Hence the `dice` is `stationary` in the *interval*, which is also the value for `translation`.
- There exist few *intervals* in all the observation *intervals* of the oracle where center of gravity of the `dice` changed its position. Therefore, definition *6.2.11* is not true (i.e. the `dice` is not `fixed`). Hence according to axiom *6.2.13*, definition *6.2.12* is true and `mobility` gets the value of `movable`.
- From axiom *6.2.23* the `dice` is either `rollable` or `dragable`. Since definition *6.2.22* can not hold true for any *interval* (i.e. the `dice` is not rollable), therefore `rollability` gets the value `nonrallable` according to axiom *6.2.23*.

The *query generator* generates following two queries for the *reasoning module* after the values of propoerties are filled in.

1. `showHyp([objectID(dice), place(floor), stability(stable), translation (stationary), mobility(movable), roll(rollable)], table, gravity)]`.
2. `showHyp([objectID(dice), place(floor), stability(stable), translation (stationary), mobility(movable), roll(rollable)], table, air)]`.

The only difference in the above *queries* is that of involved *physical phenomenon* (i.e. `gravity` and `air`). For `gravity` the *reasoning module* shows following hypotheses for the *intermediate* states of the `dice`.

1. `IntermediateState(dice):-`
   `stability(stable), translation(stationary), place(floor).`
2. `IntermediateState(dice):-`
   `stability(unstable), translation(moving), place(floor).`
3. `IntermediateState(dice):-`
   `stability(unstable), translation(moving), place(table).`
4. `IntermediateState(dice):-`
   `stability(unstable), translation(moving), place(chair).`

For `air` the *reasoning module* generates following hypotheses.

1. `IntermediateState(dice):-`
   `translation(moving), place(floor).`
2. `IntermediateState(dice):-`
   `translation(stationary), place(floor).`

In the above hypotheses, (3) (for `gravity`) represents the situation that actually occurs in figure 4.2. Let us assume that *hypotheses receiver* in figure 5.4 is able to select this *hypothesis* using some heuristics or methods. Using the *definition* component, the *hypotheses receiver* interprets this hypothesis as following,

"There exists some interval (after the release of `dice`) in which the `dice` was *on* the `table` and its center of gravity was not at a *fixed* point and the projection from its center of gravity to downward direction left the *top* of the table in at least one of the *instants* of the interval."

In this scenario, if we replace the `dice` with the `paper`, then the hypotheses formed because of `air` change. The `place` takes on all the values of possible places and the `translation` takes on both values (i.e. `moving` and `stationary`). This gives 2^n hypotheses (where n is the number of possible places for `paper`). The hypotheses generated because of `gravity` remain all the same (except that `dice` is replaced by `paper`). If we analyze this situation, it is clear that we have *redundant* information in the hypotheses for the `paper`. If `air` is the phenomenon involved in causing the fault while putting a `paper` on the table, then the robot does not really need to know about its *stability* and *translation* etc. However, for *reasoning module* both `dice` and `paper` are *objects*, which behave differently just because of their weights. Therefore, the reasoning module has to treat and evaluate both situations similarly. This can cause an exponential increase in the number of hypotheses generated by the *reasoning module*.

7.2 Scenario II and III

Before considering the application of the proposed approach to scenario II let us first consider its application to scenario III. In figure 4.4 the robot picks up a `bottle` from the `table`. Assume that the `bottle` is not correctly picked up and it falls down on the `floor`. The detection of the fault is again made through failing relations in the effects of robot's actions. This time, the *type* would be decided based on a relation like `holding(bottle)`. If the decided *type* is *location* and the `bottle` is on the `floor` in its *final state* then all the six *hypotheses* shown in previous section also hold here in similar manner. In addition to those, the *reasoning module* also finds following new hypothesis for `gravity`.

```
IntermediateState(bottle):-
stability(unstable), translation(moving), place(gripper).
```

All previous *hypotheses* hold here because we are considering the fault to be *external* which makes this situation a special case of scenario I. It can be seen that above stated *hypothesis* and *hypothesis*(3) [2] (for `gravity`) in previous section can both be very likely the actual *intermediate states* which caused the fault. However, there is one hidden problem here. The interpretations of `stability` and `place`, (according to *definitions 6.2.1* and *6.2.14*) does not specify the correct meaning of these *hypotheses*. That is, `gripper` is a `place` for the `bottle` not because the `bottle` can be placed *on* it but because the `bottle` can be held *within* `gripper`. Similarly, an *unstable* object in the `gripper` is one that is not held correctly and not the one who's C.G, leaves `gripper`'s top in some instant. This is why it is required to have different definitions for the same concept in the framework defining the properties (see observation '5' in section 6.2).

Now we change the situation in scenario III and assume that the `bottle` does not fall on the floor, but it falls on the `table` (either standing tall or on its sides). In this case the reasoner is able to generate following *hypotheses*. For `gravity`

1. `IntermediateState(bottle):-`
   `stability(stable), translation(stationary), place(table).`
2. `IntermediateState(bottle):-`
   `stability(unstable), translation(moving), place(table).`
3. `IntermediateState(bottle):-`
   `stability(unstable), translation(moving), place(gripper).`

For `air`, the *hypotheses* are same as those in previous section except that the `place` is instantiated with `table` instead of `floor`.

[2] After replacing `dice` with `bottle`

In these *hypotheses* (3) is indicating the actual reason of the fault, provided the correct definition. Hypothesis (2) also indicates a very likely situation when it is interpreted according to the definitions given in section 6.2. It can be noted here that we assume that the object in its *final state* is correctly recognized by the robot. That is, the fallen `bottle` is recognized as the same `bottle`.

Scenario II, shown in figure 4.3, is not solved in this work. This is because, according to our view the reason of the fault lies in the *shape* of the concerned objects and not in their *location*. It is expected that the fault in this scenario would be detected by the *fault detector* by some failing relation like `in(duck,basket)`. The `in/2` predicate (or a similar relation) is actually relevant to *shape type* and not to *location*. In this work we have not developed the definitions regarding the *shape* of objects because of timing constraints. Since, the *laws* regarding the shape also depend upon such definitions, it is not useful to state the laws related to shape without the meaning of definitions. A general inspection of definition related to concepts relevant to *shape* suggests that such definitions require much more geometrical information than what is used in the definitions related to *location*.

7.3 Limitations

The approach proposed in this work for fault diagnosis also has some limitations. These limitations mainly come from *naive physics* itself and the *qualitative reasoning* used in the *reasoning module*. Below we enumerate some of the significant limitations[3].

- *Difficulties with the naive physics* described in section 2.1.1 makes the development of *reasoning* rules and the *definitions* very hard. Among these difficulties '1', '3' and '4' cause significant problems in application of the proposed schema in figure 5.4 to practical scenarios.
- *Reasoning resembles rule-based reasoning* because *external faults* are caused by *physical phenomenon* and these phenomenon follow *laws/rule* of physics. This forces the reasoning to resemble rule-based reasoning which suffers from brittleness. Although in the proposed approach, the *reasoning module* reasons at a fairly high level of abstraction which significantly shrinks the size of this problem but it can not be claimed solved.

[3]We do not claim here that all these limitations are absolutely unavoidable. The main purpose of the enumeration is to critically evaluate application of NP to fault reasoning in general and using it with the proposed approach in particular.

- *External agents are not considered* in this work. This means that we have not considered presence of other agents who are allowed to manipulate the object. However, at the same time, we feel that considering presence of such an agent can cause severe problems to any approach to external fault diagnosis and our approach is not an exception.

- *Diagnosis may not be enough for updating the Model* according to the figure 4.1 if the *hypotheses receiver* interprets the *hypothesis* using the *definitions* that are developed without considering such need. Another potential limitation can be the difficulties caused in any such update because of loss of information in the abstraction of world model used for reasoning.

- *Associating correct type with faults is not simple.* It may be possible that different *types* of faults can be associated with the same failing relation. It is also possible that intuitive *type* of fault is not the correct one for reasoning purpose. For example, a sharp reader might have noticed that we claimed faults described in section 4.2.2 (scenario II) to be of *type location* until we finally disclosed in section 7.2 that in our opinion those faults are of *type shape*. This is an intentional error in this thesis to substantiate our argument.

- *Definitions of properties are time dependent.* This is because the concepts behind the *naive physics* reasoning are time dependent. Although, usage of notion of time allows us to construct definitions of properties which are object and circumstances independent. However, some of these definitions depend on more than one interval of time. In practical application this calls for storage of large amount of data.

- *There is no structure in the ontology of properties* because the properties shown in figure 5.5 represent naive concepts. This problem can also be seen in the substance schema shown in section 5.2.1. Because of this problem it can not be decided for sure that which properties are more suitable for defining *physical laws*. For example, should we consider in the gravitational law that the object falls from the table because it is movable or should we say that it is because the object is rollable or both.

- *Better hypotheses require better domain knowledge embedded in coded physical laws.* A hypothesis can be seen as a possible combination of symbolic values of relevant properties that is satisfied by the constraints of a physical law. The better these constraints represent the actual situation, the better the hypothesis is. Similarly, more relevant the properties (used in the law) are, more relevant the hypotheses are.

7.4 Assumptions

Below we enumerate some of the major assumptions made in this work. These assumptions are separate from those described in section 6.2.

- The time dependent definitions of properties assume that the robot has enough observations that it can correctly associate properties with the object in its *final state*.
- The adopted approach is similar to *microworlds* approach described in section 2.1.2. However, we do not intend to develop a competency theory. The intention here is to use the theory as a cognitive model, therefore we assume that the *definitions* can contain beliefs that are plain wrong, but useful in reasoning.
- We assume that the cognitive model used for definition has its scope limited to the robot using it. This means that the definitions used by the robot are only valid for the robot in fault reasoning. These definitions or laws are not true depiction of universal laws and properties.
- The reasoning module uses few relations in its model which must be created at the time of *final state* of the object (e.g. `near(table, chair)`). Without such relations the resulting *hypotheses* can be too many. We assume that the robot possesses enough information and ability to fill in the correct values of these relations at the time of *final state*.
- The approach assumes that the robot is always able to see the object in its *final state*.

8 Related Work

In this work we have applied *naive physics* knowledge for reasoning about external faults of the robot. Although, it may appear that robotic fault diagnosis approaches are very relevant to this work, but the fact that we deal only with external faults makes this work very different from such approaches. Reasoning about external faults is more related to everyday commonsense reasoning than usual model based fault reasoning for a system or a process. For such reasoning, *rule-based reasoning* used in *expert systems* has a close relevance, where the expert system is designed to reasons about the fault based on rules which are *physical laws*.

Since we use the naive physics knowledge for reasoning and the original concept of naive physics (Hayes [1979]) centers around *formalization* of everyday knowledge, therefore the works in *knowledge representation* are related to ours, in general. Davis [1998] presents one important approach of *microworlds* for knowledge representation and reasoning. In his work, the author proposes to develop specific competency theories powerful enough to justify commonsense inferences. This approach is related to ours, in the sense that we also develop specific theories in the form of (so-called) framework(s). We already have given a brief review of this work in section 2.1.2.

The *reasoning module* in our approach uses properties of objects which are relevant to space and shape. Many formalisms have been developed for qualitative spatial representation and reasoning. A comprehensive review of these approaches can be found in Cohn and Hazarika [2001]. However, any ready-made import of such formalisms is not useful for our approach. This is because we define the properties of the object in a manner that reasoning using them reveals useful information about the fault. Our approach does not require a complete formalism that can be used for reasoning as done by engineers and scientists. It requires a framework that is representative of knowledge of physics novices, good enough to result in useful inferencing. Davis has developed a logical framework for solid objects in Davis [1986] and Davis [1988]. Although, these works use almost entire machinery of Newtonian mechanics in the framework but they are relevant to our work because we use similar primitives for our framework.

What properties are used by physics novices to reason about daily life phenomenon? and how do they use them for reasoning? These are important questions that need to be answered when developing the ontology of properties used in our approach. A survey of studies relevant to these questions is given in Reiner et al. [2000]. Authors of that work also present the *substance schema* which we use in this work. To extend our work or further refine it, useful insights can be gained from Reiner et al. [2000].

There are few other works in *qualitative reasoning* that can be associated with fault diagnosis using naive or qualitative physics, however most of these works use algorithmic approaches which we find infeasible for real world environment. A detailed discussion about such related works is not possible here. Interested reader can find a comprehensive collection of such works in Weld and Kleer [1990]. QPT and QSIM, mentioned in section 2.2 of the thesis, are two of such works.

9 Conclusion and future work

In this work we have presented an approach for fault reasoning based on *naive physics*. We consider only the *external* faults that occur in the absence of any external agent. The presented approach is evaluated by considering its application to three simple scenarios which involve manipulation tasks. Based on the evaluation we enumerate major limitations and assumptions of this work. The mechanism to reason about the faults is developed after studying major approaches in *naive/qualitative physics*. A critical review of these approaches is given in chapter 2 of the thesis. Our proposed mechanism uses *qualitative reasoning* for reasoning about faults. Such reasoning requires use of constant symbols and primitive relations which are understood by the reasoner. We have also presented a *framework* that gives definitions of the used symbols and relations. The presented *framework* is developed with an intent to show that how the definitions of the symbols should be perceived such that *naive physics* concepts behind them can be captured in a manner useful for our approach.

Application of *naive physics* knowledge for fault reasoning in real world situations is not straight forward. The main reason for that is, the laws/rules/properties utilized by a physics *novice* to reason about a daily life phenomenon is object and circumstances dependent. Usually, the person utilizes only that information (i.e. laws or properties) for reasoning which is relevant to the given circumstances. It can also not be guaranteed that the person has reached to the correct conclusion using the concepts which are consistent. There is also no real hierarchical structure in *properties* of the object that are used for reasoning. Such object/situation dependence and lack of structure in *naive physics* knowledge makes it impossible to develop an approach that is generic and algorithmic like QPT or QSIM. We find that any approach utilizing (only) QDEs for external fault reasoning is also not feasible in general, because use of QDEs to correctly predict the extended behavior of objects in real world is inadequate. Furthermore, QDEs become a very inefficient approach for the situations considered in scenarios in section 4.2 where behavior of the object is easy to be characterize over extended time than local time.

Considering the challenges stated in previous paragraph and at different places in the thesis, our proposed approach resembles microwrolds approach. We find it more feasible to use *qualitative reasoning* at a very high level of abstraction such that the symbols and relations used for reasoning are defined in a theory (i.e. a framework) that is specific to a particular *type* of fault. This means, we propose to develop different microworld theories for different types of faults. The definitions used in such theories are mutually

consistent, but we allow them to be even plain wrong. The definitions utilize only the relevant information that helps in inferencing in the reasoning module even with minimal information. The proposed theories, have their scope limited only to the system (i.e. the robot) for which they are developed and their mutual interaction is only allowed in reasoning module (where necessary) through physical laws. We find that it is feasible to use rule-based reasoning for external faults since it is natural to state physical laws (which cause the fault) as rules. The brittleness of this approach can be reduced significantly by using a very high level of abstraction for the rules.

In this work the presented approach is a *proof of concept* that utilizing *naive physics* for external fault reasoning is useful. This work does not investigate extensive application of the approach on different *types* of faults because of time constraints. We also do not claim completeness of the proposed framework of definitions in section 6.2. A possible extension of this work is to develop different frameworks (i.e. definitions) for different types of faults and apply them to real world scenarios using further physical laws. It can be noted that we expect side by side development of the *reasoning module* and the *definition* component shown in figure 5.4. Although, further development of this work using same approach can be well guided by the insights from this work, however it can not be claimed that these insights will ease the level of hardness of development on the same foot steps.

Utilizing the approach for a real robot

In the text below we briefly summarize that how our approach can be utilized for a real robot effectively. We do it by utilizing the insights from this work and assuming that we have to work on it from the scratch.

Firstly, it can be noticed that our proposed mechanism or approach has its scope limited to a particular (type of) robot. That is, the developed theories, ontologies and physical laws are developed for a particular robot and can be exported (with minimum changes) only to the robots with similar capabilities. Therefore, to use this approach for external fault reasoning in real robot, we first need to understand the capabilities (i.e. performable tasks) of the robot. An important component of this understanding is the knowledge of planning operators and methods used in the robot. Based on this understanding, the *types* of faults must be developed. Then the ontology of the *properties* is to be developed. It should be noticed that there is no correct or wrong ontology for properties. The ontology used in this work is derived form the substance schema shown in section 5.2.1. It is expected that this ontology can vary greatly if the types of faults are different. For each type of fault, a pool of definitions (i.e. framework) is to be developed. In our opinion development of definitions can follow the same approach as this work. However, it may be possible to utilize definitions more effectively by (somehow) letting the *oracle* know few permanent objects and their intrinsic properties. For example, if the *oracle* already knows

the names of `fixed` objects in its environment then there is no need to define `movable` and `fixed` etc..

In this work we let the *hypotheses receiver* to interpret the hypothesis using the *definition* component that was also used to determine the *final state* of the object. A more effective approach would be to let the *hypotheses receiver* interpret the hypothesis on different definitions, that are developed based on the knowledge of the *model* used for *prediction* in figure 4.1. Although maintaining the correct interpretation of the properties separately is more work but it can have two major advantages.

1. *Updating* the *model* (in figure 4.1) would become much easier and effective.
2. The *definitions* used for determining the final state can be based on further less knowledge.

Appendix

A: Definitions

Below are some important definitions (found in the literature regarding fault diagnosis and naive physics) of the terms used in the thesis.

Fault
An unpermitted deviation of at least one characteristic property of parameter of the system from the acceptable/usual/standart condition.

Fault detection
Determination of the presence of fault in the system and the time of detection.

Fault isolation
Determination of kind, location and time of detection of a fault.

Fault identification
Determination of the size and time-variant behavior of a fault.

Fault diagnosis
Determination of the kind, size, location and time of detection of the fault.

Metric Diagram
A combination of symbolic and quantitative information used as an oracle for a class of spatial questions. (Definition is specific to Forbus et al. [1991].)

Mereology
Mereology is the theory of parthood relations: of the relations of part to whole and the relations of part to part within a whole. (from stanford encyclopedia of philosophy)

Ontology
Ontology concerns how to carve up the world, i.e., what kinds of things there are and what sorts of relationships can hold between them Forbus [2003].

Place Vocabulary
A purely symbolic description of shape and space, grounded in the metric diagram. (Definition is specific to Forbus et al. [1991].)

Qualitative behavior
A sequence of qualitative states occurring over a particular span of time is called a behavior Forbus [2003].

Qualitative Differential Equation Model
A qualitative differential equation model (QDE) is an abstraction of an ordinary differential equation, consisting of a set of real-valued variables and functional, algebraic and differential constraints among them Kuipers [1986].

Qualitative state
A qualitative state is a set of propositions that characterize a qualitatively distinct behavior of a system Forbus [2003].

Quantity space
A set of ordinal relationships that describes the value of a continuous parameter Forbus [1984].

B: Query generator code

```
%%%% Query generator code in Prolog.
%%%% This code was used to achieve results reported here.
%%%% The code is written so that it is easily extendable just by
%%%% entering new fault 'types' or 'properties' using same approach

%%%% Process called when fault occurs
  fault(FaultType):-
    subfaultCall(FaultType, SubFaultList),
    faultProperties(FaultType, SubFaultList).

%%%% Takes in the values for the associated properties of each fault and
     writes the Query to be asked on the terminal.

faultProperties(FaultType, SubFaultList):-
    write('Enter␣values␣for␣following␣properties␣from␣the␣final␣state:')
       ,
    nl,
    propertiesCall(FaultType, PropertyList),
    enumerate(PropertyList,FinalStateList),
    expectedProperty(FaultType, Property),
    generateQuery(FinalStateList, SubFaultList, Property).

generateQuery(_,[],_):-nl,!.
generateQuery(FinalStateList, [SubFaultHead|SubFaultTail],
   ExpectedProperty):-
nl,write('showHyp('),write(FinalStateList),write(','),write(
   ExpectedProperty),write(','),write(SubFaultHead),write(').'),nl,
   generateQuery(FinalStateList, SubFaultTail, ExpectedProperty).
```

```
%%%%%% Calls for query generation %%%%%%%%

%%%%%%% Subfault calls%%%%%%%
subfaultCall(location, L):-
    L = [gravity,air].
subfaultCall(movement, L):-
    L = [displacement, stoppage].
subfaultCall(shape,L):-
    L = [distortion].

%%%%%%% Expected property%%%%%%
expectedProperty(location,Property):-
    write('expected␣place'),nl,
    read(Property).

%%%%%%% Properties calls%%%%%
propertiesCall(location, L):-
    L = [objectID, stability, roll, mobility, translation, place].

propertiesCall(movement, L):-
    L = [objectID , mobility].

propertiesCall(shape, L):-
    L = [objectID , transformable, container, stickable].

%%%%%%% Auxiliary predicates%%%%%%%%
writeList([]).
writeList([X|L]):-
    write(X),nl,
    writeList(L).

concatenate([], L, L).
concatenate([X1|L1],L,[X1|L2]):-
    concatenate(L1, L, L2).

addTerm(X, L, [X|L]).

makeTerm(X,Y,Term):-
    Term =..[X,Y].

enumerate([],[]).
enumerate([H|T], [H1|T1]):-
    write(H), write('='),nl,
    read(Property),
    makeTerm(H, Property, H1),
    enumerate(T,T1).
```

C: Reasoning module code

```
%%%%% Model of the world %%%%%%%%

%%%% Involved objects %%%%%
object(table).
object(dice).
object(floor).
object(bottle).
object(basket).
object(bottle).
object(paper).
object(shelf).

%%%%% Numeric values of properties %%%%%

%%height of object in centimeters
height(table, 80).
height(dice, 10).
height(bottle, 20).
height(basket, 60).
height(duck, 7).
height(floor, 0).
height(chair, 50).
height(shelf,150).
height(gripper, 100).

%%Weight of the objects in grams
weight(table, 7000).
weight(dice, 50).
weight(bottle, 100).
weight(basket, 2000).
weight(duck, 20).
weight(paper,1).
weight(shelf, 8000).

%%%%%% Possible values of properties %%%%%%

place(floor).
place(table).
place(chair).
place(shelf).
place(gripper).

drag(dragable).
drag(nondragable).
```

```
roll(rollable).
roll(nonrollable).

mobility(moveable).
mobility(fixed).

stability(stable).
stability(unstable).

translation(moving).
translation(stationary).

roll(rollable).
roll(nonrollable).

containability(containable).
containability(uncontainable).

%%%%%% Relations between objects

near(place(X), place(X)).                    %%everyplace is near itself.
near(object(X), object(X)).                      %%every object is near
    itself.

near(place(table),place(chair)).
near(place(chair),place(table)).
near(place(gripper), place(chair)).
near(place(chair), place(gripper)).
near(place(gripper), place(table)).
near(place(_),place(floor)).

%%%%%% For generating output in prolog syntax %%%%%%%%%
showHyp(State, ExpectedProperty, Law):-
    nl,
    applyLaw(State, ExpectedProperty, Law).

showClause([H|T]):-
    write(H),
    (T = [],nl,write('The␣hypothesis␣could␣not␣be␣determined');
    write(':-'),nl),
    tab(2),
    showTail(T).

showTail([]):-
    write('.').
showTail([H|T]):-
```

```
    write(H),
    (T = [],!, write('.'),nl
    ;
    write(','),
    tab(1),
    showTail(T)).

%%%%%%%%%%%%%%% Processing of query %%%%%%%%%%%%%%%%%%%%%%%%%

applyLaw(FinalState, ExpectedLocation, gravity):-
    gravityTransitions(FinalState, ExpectedLocation,  L),clauseForm(L),
       nl.

applyLaw(FinalState, ExpectedProperty, air):-
    airTransitions(FinalState,ExpectedLocation, L),clauseForm(L),nl.

%applyLaw(FinalState, ExpectedProperty, adhesion):-
%applyLaw(FinalState, ExpectedProperty , displacement):-
%applyLaw(FinalState, ExpectedProperty , stoppage):-
%applyLaw(FinalState, ExpectedProperty , distortion):-


%%%%%%%%%%%%%%% Physical Transitions %%%%%%%%%%%%%%%%%%%%%%%%%

gravityTransitions([objectID(ID1), stability(St1), roll(R1),  mobility(
   movable), translation(T1), place(P1)], ExpectedLocation, [objectID(
   ID1), stability(St2),translation(T2), place(P2)]):-
     place(P2), place(P1), height(P1, H1), height(P2, H2), height(
        ExpectedLocation, H3), near(place(ExpectedLocation), place(P2)),
         (smaller(H1, H2), St2 = unstable, T2 = moving; not(smaller(H1,
        H2)), (H1 = H2), (St2 = stable, T2 = stationary; St2 = unstable,
         T2 = moving)).


airTransitions([objectID(ID1), stability(St1), roll(R1),  mobility(
   movable), translation(T1), place(P1)], ExpectedLocation, [objectID(
   ID1), translation(T2), place(P2)]):-
    translation(T2), place(P1), place(P2), weight(ID1,W), (smaller(W,5),
        different(P1,P2);smaller(5,W),same(P1,P2)).

%%%%%%%%%%%%%%% Making clause of state%%%%%%%%%%%%%%%%%%%%
clauseForm([H|L]):-
    H = objectID(Obj),
    makeTerm(['IntermediateState', Obj], Head),
    addTerm(Head, L, X),
    showClause(X).
```

```
%%%%%%%%%%%%%% Auxiliary definitions %%%%%%%%%%%%%%%%%%%%%%%%
makeList([],[]).
makeList([H1|T1], [H2|T2]):-
    makeTerm(H1,H2),
    makeList(T1,T2).

makeTerm(L, Z):-
    Z =..L.
addTerm(X, L, [X|L]).

same(X,Y):-
    X = Y.

different(X,Y):-
    not(same(X,Y)).
smaller(X,Y):-
    X<Y.
```

D: Use cases description

Detect fault (UC1): Detects the fault when effects of the last action of the robot are not achieved. It gets triggered when the action has been performed. In case the action is successful the robot is informed about that. When the fault is detected, this use case also specifies the *type* of the fault. It is assumed that the use case is always able to associate relevant *type(s)* to the fault.

Trigger reasoning (UC2): When the fault has been detected, this use case triggers the *query generation* process. The trigger signal is compatible to the *query generator* and it contains the *type* of the fault.

Receive hypotheses (UC3): The *hypotheses receiver* receives a list of hypotheses through this behavior. The use case is triggered after UC2 has sent the request to the *query generator*. UC3 listens to any signal from UC13, this signal consists of a list of hypotheses. After receiving the list this use case is ready to interpret the received hypothesis. It is assumed that this use case always receives a list of hypotheses from UC13.

Interpret hypotheses (UC4): This use case interprets the hypotheses for the *hypotheses receiver*. It starts after UC3 has successfully received the hypotheses list. UC4 extends UC5 to interpret the properties used in the hypotheses. As a result of this use case the hypotheses receiver is able to know the meaning of each hypothesis received in UC3.

Define properties (UC5): Knowledge base defines the properties used by UC4 and UC8. This use case depicts the logical representation of concepts in the knowledge base.

Create world model (UC6): The *oracle* creates/updates the model of the world after UC1 has detected the fault. This use case results into an updated model of the world after the object has reached its *final state*. It is assumed by the use case that the observer has the compete information about the world at the time of creation of the world model.

Observer world (UC7): This behavior related to the observer is continuous. However, the observer uses the information of observations only after the object has reached its final state.

Define final state (UC8): This use case defines final state of the object after UC9 receives the trigger signal from UC2. It extends UC5 to recognize the relevant properties of the object in its final state. The oracle knows the object in its final state because of the same assumption that the robot posses the complete knowledge of the object in its final state. The effect of this behavior is the availability of completely specified final state of the object.

Receive reasoning trigger (UC9): After UC1 has detected the fault, this use case listens to any signal from UC2. After it receives such a signal the *query generator* uses it to initiate UC8. It is assumed that any possible signal produced by UC2 is compatible with UC9.

Generate query (UC10): It generates the relevant query to be received by UC11. The generation of the query is the result of association of relevant properties to respective physical laws. UC10 initiates after UC8 has finished its work.

Receive query (UC11): This use case listens to the signal from UC10 after any fault is detected by UC1. It assumes that UC10 generates only those signals which are compatible for reasoning in UC12. A list of queries is ready to be processed by UC12, after UC11 has finished its job.

Perform reasoning (UC12): This use case signifies the reasoning process. It processes each query of the signal from UC11. For each query it generates a list of reasons that could cause the fault. The reasoning process also includes the world model creation, in which it models some relations between objects and assign some numerical values to few properties. This use case assumes that the observer posses enough information that the relevant relations and properties get correct values at the time of final state of the object.

Generate hypotheses(UC13): It makes the reasons of faults found by UC12 compatible to be used by UC4. The effect of this use case is a list of hypotheses ready to be interpreted by UC4.

CD Content

- This document as PDF
- Prolog code for
  - Reasoning module
  - Query generator

Bibliography

Sobhi Baniardalani, Javad Askari, and Jan Lunze. Qualitative model based fault diagnosis using a threshold level. *International Journal of Control, Automation, and Systems*, 2010. 10

Ivan Bratko. *PROLOG Programming for Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 2001. ISBN 0201416069. 6

Jennifer Carlson and Robin R. Murphy. Reliability analysis of mobile robots. In *ICRA*, pages 274–281, 2003. 10

A. G. Cohn and S. M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundam. Inf.*, 46(1-2):1–29, 2001. 8, 39

Matthew J. Daigle. *A qualitative event-based approach to fault diagnosis of hybrid systems.* PhD thesis, Graduate School of Vanderbilt University, 2008. 11

Earnest Davis. A logical framework for solid object physics. Technical Report 245, New York University Computer Science Department, October 1986. 26, 39

Ernest Davis. A logical framework for commonsense predictions of solid object behaviour. *AI in Engineering*, 3(3):125–140, 1988. 39

Ernest Davis. The naive physics perplex. *AI Magazine*, 19:51–79, 1998. 4, 5, 6, 39

J de Kleer and B C Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, 1987. ISSN 0004-3702. 10

Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24(1-3):85–168, 1984. ISSN 0004-3702. 7, 45

Kenneth D. Forbus. *Qualitative Reasoning.* MIT Press, 2003. 6, 8, 44, 45

Kenneth D. Forbus, Paul Nielsen, and Boi Faltings. Qualitative spatial reasoning: The clock project. *Artif. Intell.*, 51(1-3):417–471, 1991. 8, 25, 44

P. J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Micro Electronic Age*, pages 242–270. Edinburgh University Press, 1979. 3, 4, 39

P. J. Hayes. *The second naive physics manifesto*, pages 46–63. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-095-7. 4, 5

George M. Coghill Honghai Liu. A model-based approach to robot fault diagnosis. *Knowledge-Based Systems*, 2005. 10

Robert Kowalski. Algorithm = logic + control. *Commun. ACM*, 22:424–436, July 1979. ISSN 0001-0782. 4

Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986. 7, 45

Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting people's common sense into knowledge bases of household robots. In *33rd Annual German Conference on Artificial Intelligence (KI 2010)*, Karlsruhe, Germany, September 21-24 2010. Springer. 11

Lars Kunze, Mihai Dolha, Emitza Guzman, and Michael Beetz. Simulation-based temporal projection of everyday robot object manipulation. In Yolum, Tumer, Stone, and Sonenberg, editors, *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Taipei, Taiwan, May, 2–6 2011. IFAAMAS. 11

H. LIU, G. M. COGHILL, and H.Y. XU. Qualitative modelling of kinematic robots for fault diagnosis. *International Journal of Production Research*, 2005. 11

John McCarthy. Programs with common sense. In *Semantic Information Processing*, pages 403–418. MIT Press, 1968. 4

A. Monteriu, P. Asthana, K. P. Valavanis, and S. Longhi. Real-time model-based fault detection and isolation for ugvs. *J. Intell. Robotics Syst.*, 56:425–439, November 2009. ISSN 0921-0296. 10

R. Moore. The role of logic in knowledge representation and commonsense reasoning. In *Second national conference on artificial intelligence. Melo Park, Calif: American Association for Artificail Intelligence*, 1982. 4

Ron J. Patton, Paul M. Frank, and Robert N. Clarke, editors. *Fault diagnosis in dynamic systems: theory and application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-13-308263-6. 10

O. Pettersson, L. Karlsson, and A. Saffiotti. Model-free execution monitoring in behavior-based robotics. *IEEE Trans. on Systems, Man and Cybernetics.*, 37(4):890–901, 2007. 10

P. M. Frank R. J. Patton. *Issues for fault diagnosis of dynamic systems*. Springer-Verlag, 2000. 12, 13

M. Reiner, J.D. Slotta, M.T.H. Chi, and L.B. Resnick. Naive physics reasoning: A commitment to substance-based conceptions. *Cognition and Instruction*, 2000. iii, 4, 5, 22, 39

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002. ISBN 0137903952. 3, 8

Jochen Schroder. *Modelling, State Observation, and Diagnosis of Quantised Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. 10

V. Venkatasubramanian. A review of process fault detection and diagnosis part ii, qualitative models and search strategies. *Computers and Chemical Engineering*, 27(3):313–326, March 2003. 10

Vandi Verma and Reid G. Simmons. Scalable robot fault detection and identification. *Robotics and Autonomous Systems*, 54(2):184–191, 2006. 10

Vandi Verma, Geoffrey Gordon, Reid Simmons, and Sebastian Thrun. Real time fault diagnosis robot fault diagnosis. *Robotics and Automation Magazine*, 11(1):56 – 66, June 2004. 10

S. Vosniadou. On the nature of naive physics. In *M. Limon & L. Mason (Eds). Reframing the process of conceptual change*. Kluwer academic publisher, 2002. 5

D. S. Weld and J. De Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufman, San Mateo, Ca, 1990. 8, 40